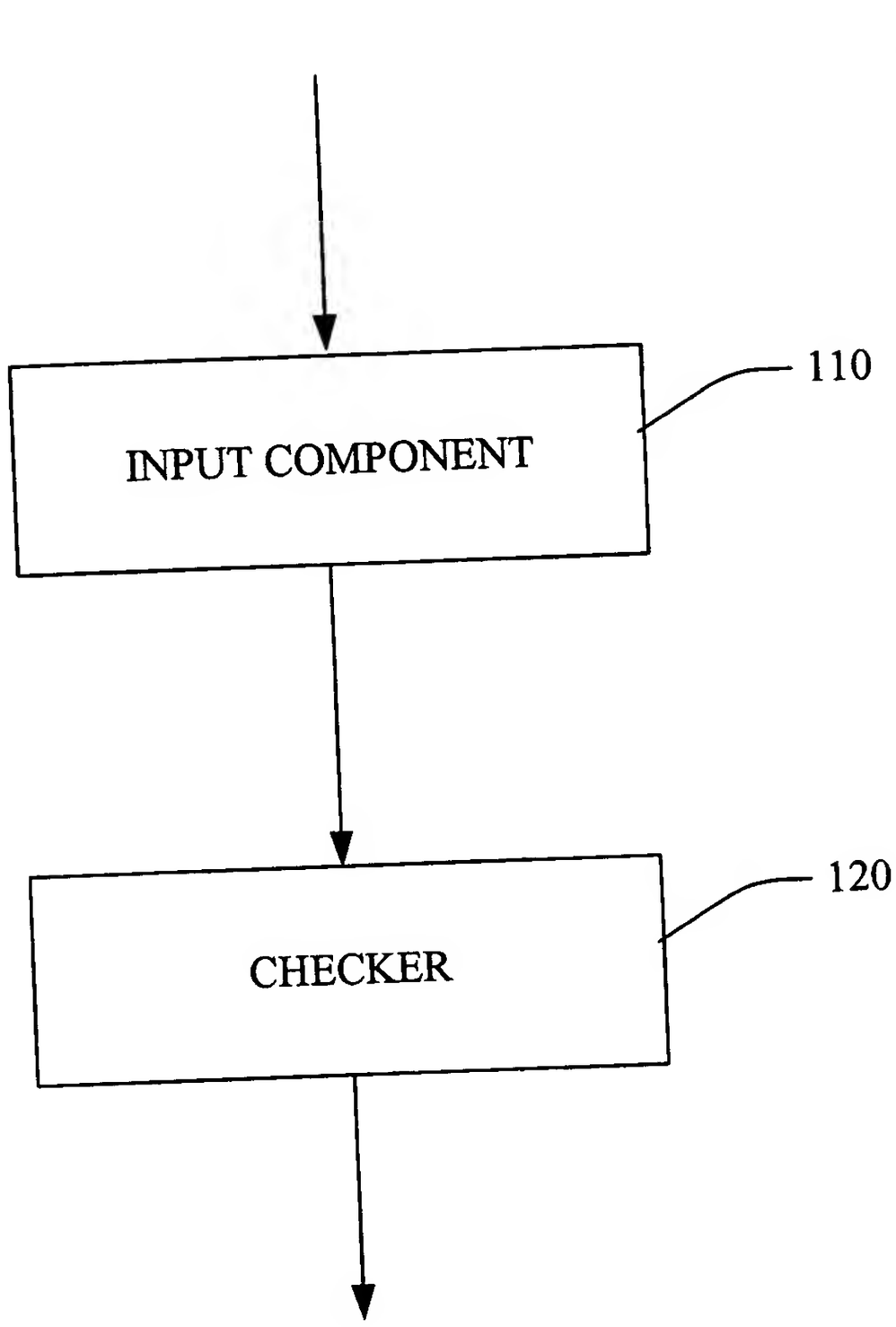

FIG. 1

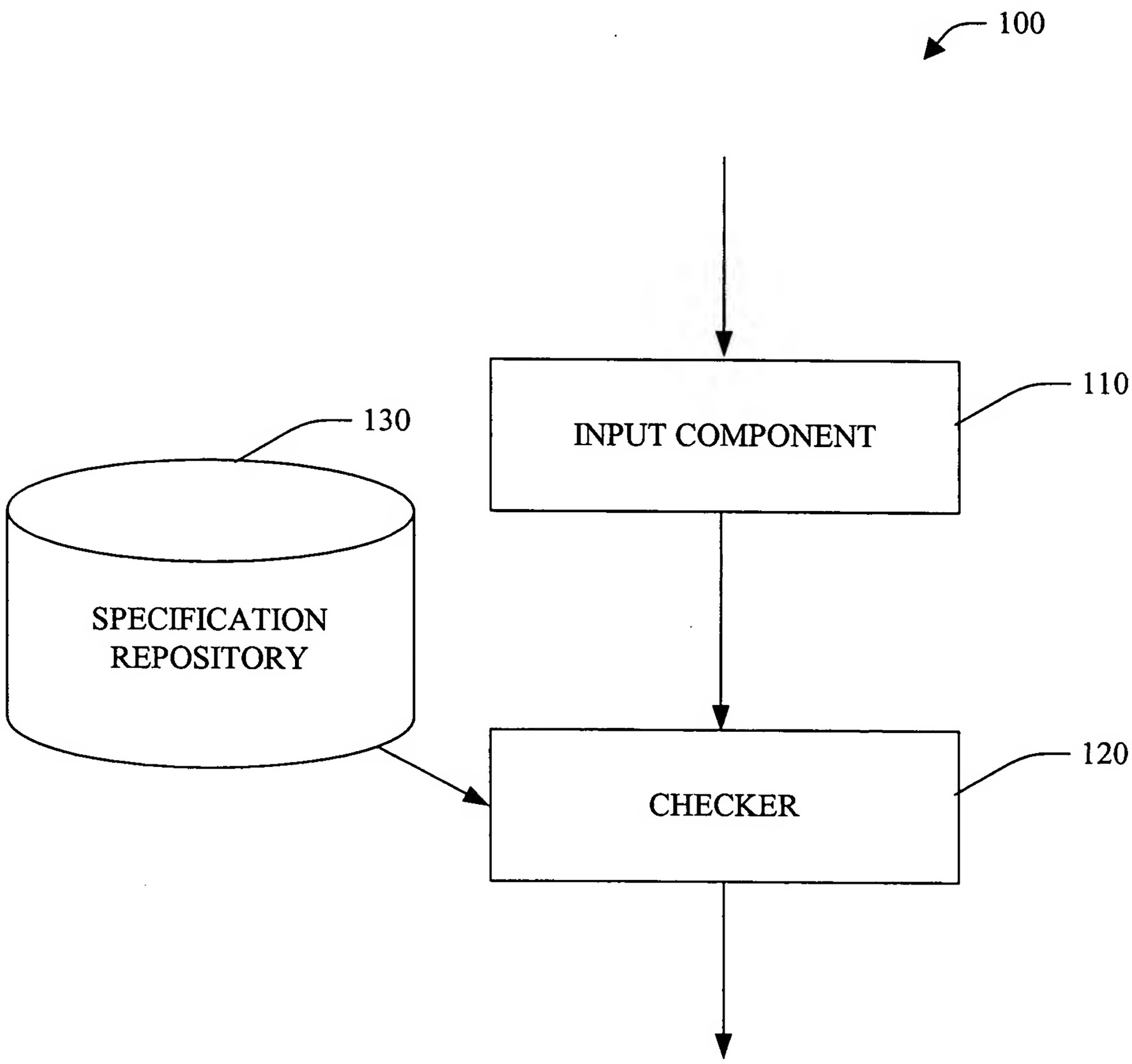
100
INPUT COMPONENT
110
130
SPECIFICATION
REPOSITORY
CHECKER
120

FIG. 2

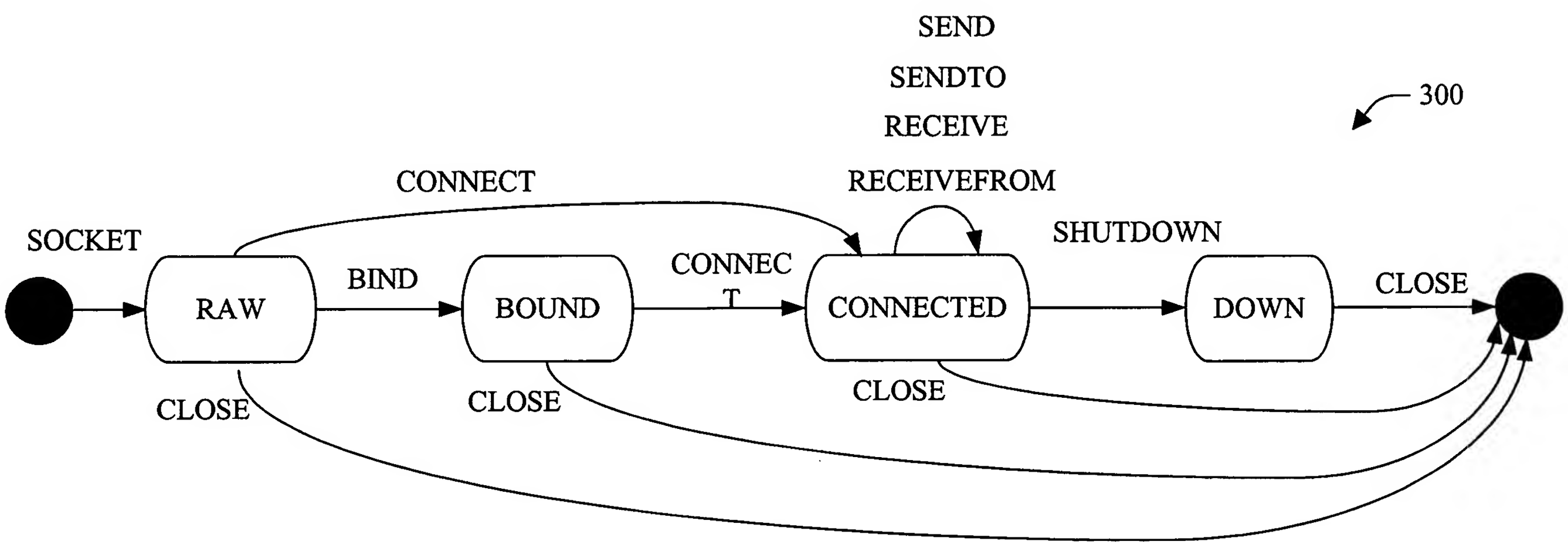
300
SEND
SENDTO
RECEIVE
RECEIVEFROM
CONNECT
SOCKET
SHUTDOWN
BIND
CONNEC
T
CLOSE
RAW
BOUND
CONNECTED
DOWN
CLOSE
CLOSE
CLOSE

FIG. 3

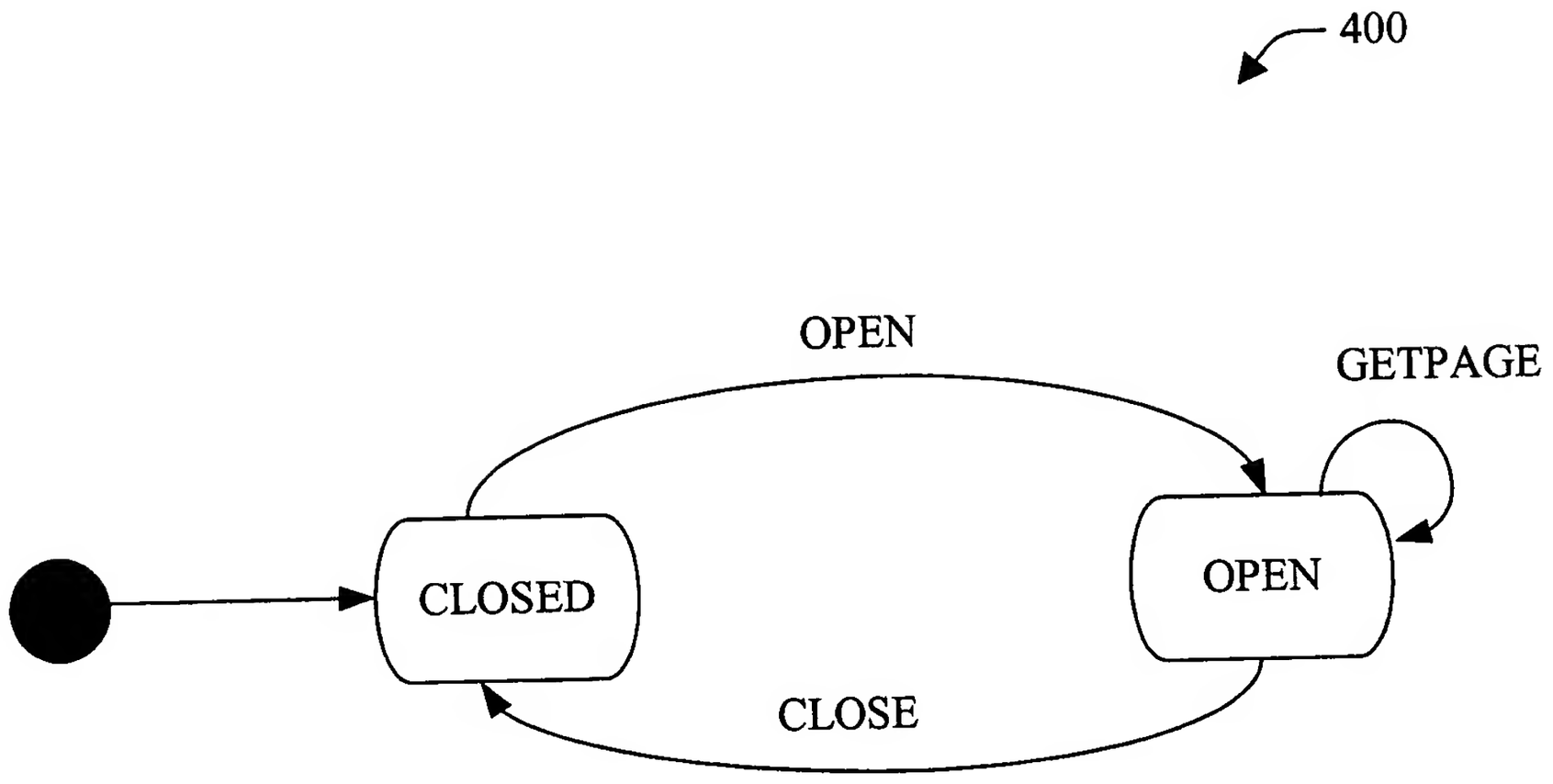

FIG. 4

500

```
[WithProtocol(
        CustomState=typeof(SqlConnectionState)) ]
class SqlConnection
{
[ Creates,
 OutConnectionState(
        Status=ConnectionState.Closed,
        Host="", Database="")]
SqlConnection ();

[ Creates,
 OutConnectionState(
        Status=ConnectionState.Closed,
        StateProvider="NewHostAndDatabase"),
OutStateDependsOn("connectionString")]
SqlConnection (string connectionString) ;

[ OutConnectionState(
        Status=ConnectionState.Open) ]
void Open () ;
}
```

FIG. 5

600

```
[ WithProtocol(
        CustomStat=typeof(SqlCommandState)) ]
class SqlCommand
{
  [OutCommandState(
        StateProvider="UpdateCommandText"),
   OutStateDependsOn("cmdText") ]
  SqlCommand (string cmdText);

  [ property: Transparent ]
  SqlConnection Connection { get; set; }

  [ InCommandState(
        StateChecker="CheckCommandText"),
   InStateDependsOn("this.Connection") ]
  [ return: OutReaderState(
        StateProvider="GetColumnInfo"),
   OutStateDependsOn("this.Connection","this") ]
  SqlDataReader ExecuteReader ();
}
```

FIG. 6

700

```
{ WithProtocol(
        CustonState=typeof(sqlReaderState)) ]
class SqlDataReader
{
        [ InReaderState(
          StateChecker="ValidColumnName"),
           InStateDependsOn("name") ]
        object get_Item (string name);

        [ InReaderState(
          StateChecker="ColumnIsString"),
           InStateDependsOn("i") ]
        string Getstring (int i);
}
```

FIG. 7

800

```
class SqlConnectionState : CustomState
{
        ConnectionState Status
        sting Host, Database;

        void NewHostAndDatabase (string{} connString) {
        // Example plug-in postcondition, which
        // parses a connection string for
        // its host and database names.
        Regex hostRegex = new Regex (
            @"(data source|server)\s*=([^;]*)\b",
            RegexOptions.IgnoreCase);
    Regex dbRegex = new Regex(
            @"(catalog|database)\s*=([^;}*\b",
            RegexOptions.IgnoreCase);
    for (int i=0; i<connString.Length; i++) {
            MatchCollection dbm =
              hostRegex.Matches(connString[i]);
            if (dbm.Count > 0)
               Host = dbm[0] .Groups[2].Captures[0].Value;
            MatchCollection hm =
               dbRegex.Matches(connString[i]);
            if (hm.Count > 0)
               Database = hm[0].Groups[2].Captures[0].Value;
    }
    if (Host == null)
            Fail("could not find host");
    if (Database == null)
            Fail("could not find database");
        }
}
```

FIG. 8

900

```
class SqlCommandState : CustomState
{
        string[] CommandText;

        void UpdateCommandText (string[] c0 { CommandText=c; }

        bool CheckCommandText (SqlConnectionState c) {
            return ISLegalSQL(CommandText, c.Host, c.Database);
        }
}
```

FIG. 9

1000

```
class SqlReaderState : CustomState
{
        string [] ColumnNames, ColumnTypes;

        void GetColumnInfor (SqlConnectionState connection,
                             SqlComandState command) {...}
        bool ValidColumnName (string[] name) {...}
        bool ClumnIsString (int i) {...}
}
```

FIG. 10

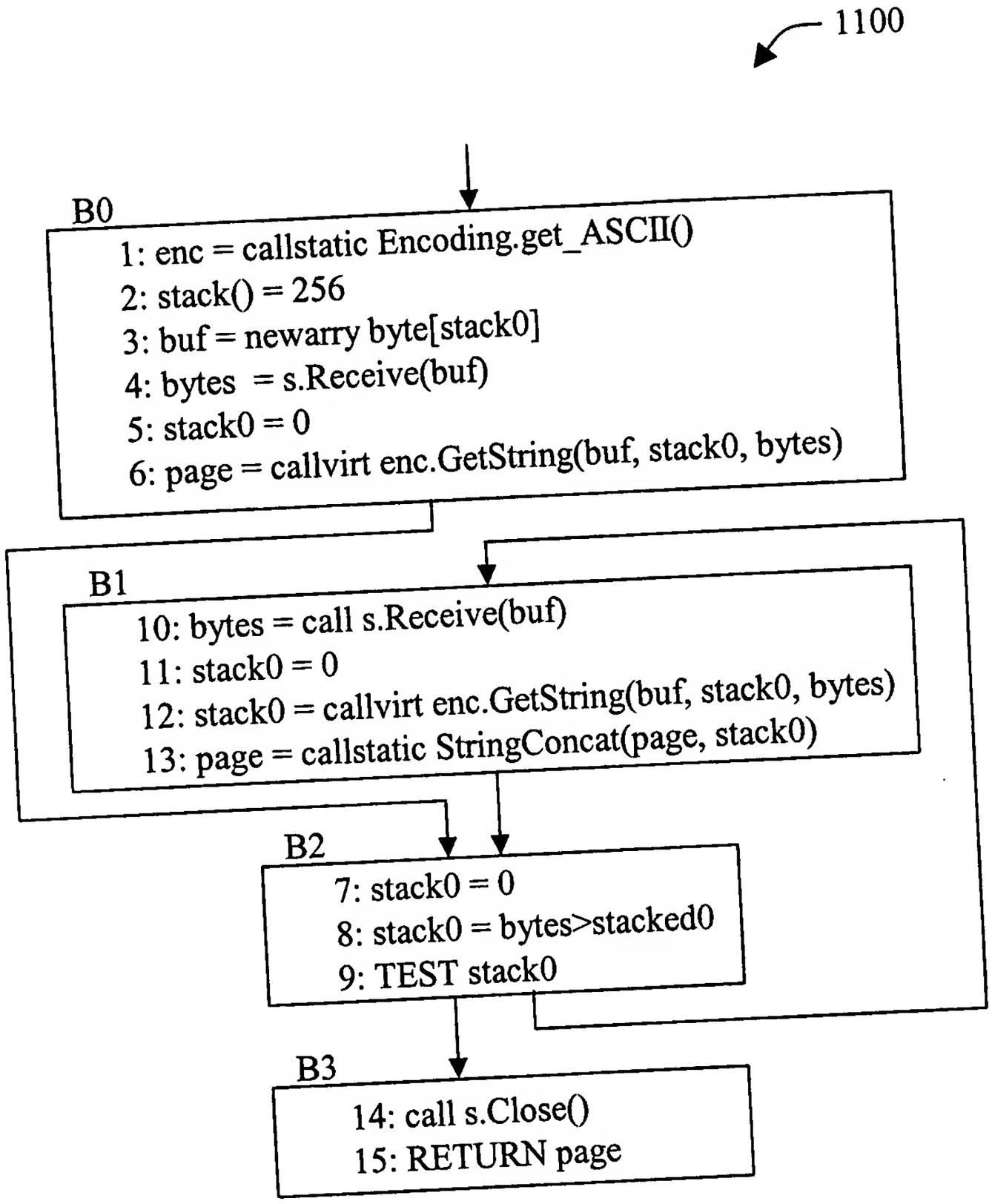
1100
B0
1: enc = callstatic Encoding.get_ASCII()
2: stack() = 256
3: buf = newarry byte[stack0]
4: bytes = s.Receive(buf)
5: stack0 = 0
6: page = callvirt enc.GetString(buf, stack0, bytes)
B1
10: bytes = call s.Receive(buf)
11: stack0 = 0
12: stack0 = callvirt enc.GetString(buf, stack0, bytes)
13: page = callstatic StringConcat(page, stack0)
B2
7: stack0 = 0
8: stack0 = bytes>stacked0
9: TEST stack0
B3
14: call s.Close()
15: RETURN page

FIG. 11

1200

0	s : ref(a_0)	a_0 → (Socket.NotAliased,"connected", 0)
1	enc : ref(a_1)	a_1 → (Encoding.MayBeAliased/Escaping, default, 0)
2	stack0 : calue(int, 256, default)	
3	buf : value(byte[],·, default)	
4	bytes : value(int, ·, default)	
5	stack0 : value(int, 0, default)	
6	page : ref(a_3)	a_3 → (string.MayBeAliased/Escaping,default, 0)
7	stack0 : value(int, 0, default)	
8	stack0 : value(bool, ·, default)	
9	(no change)	
10	bytes : value(int, ·, default)	
11	stack0 : value(int, 0, default)	
12	stack0 : ref(a_4)	a_4 → (string.MayBeAliased/Escaping,default, 0)
13	(no change)	(a_0 removed from capabilities)
14		
15	(no change)	

FIG. 12

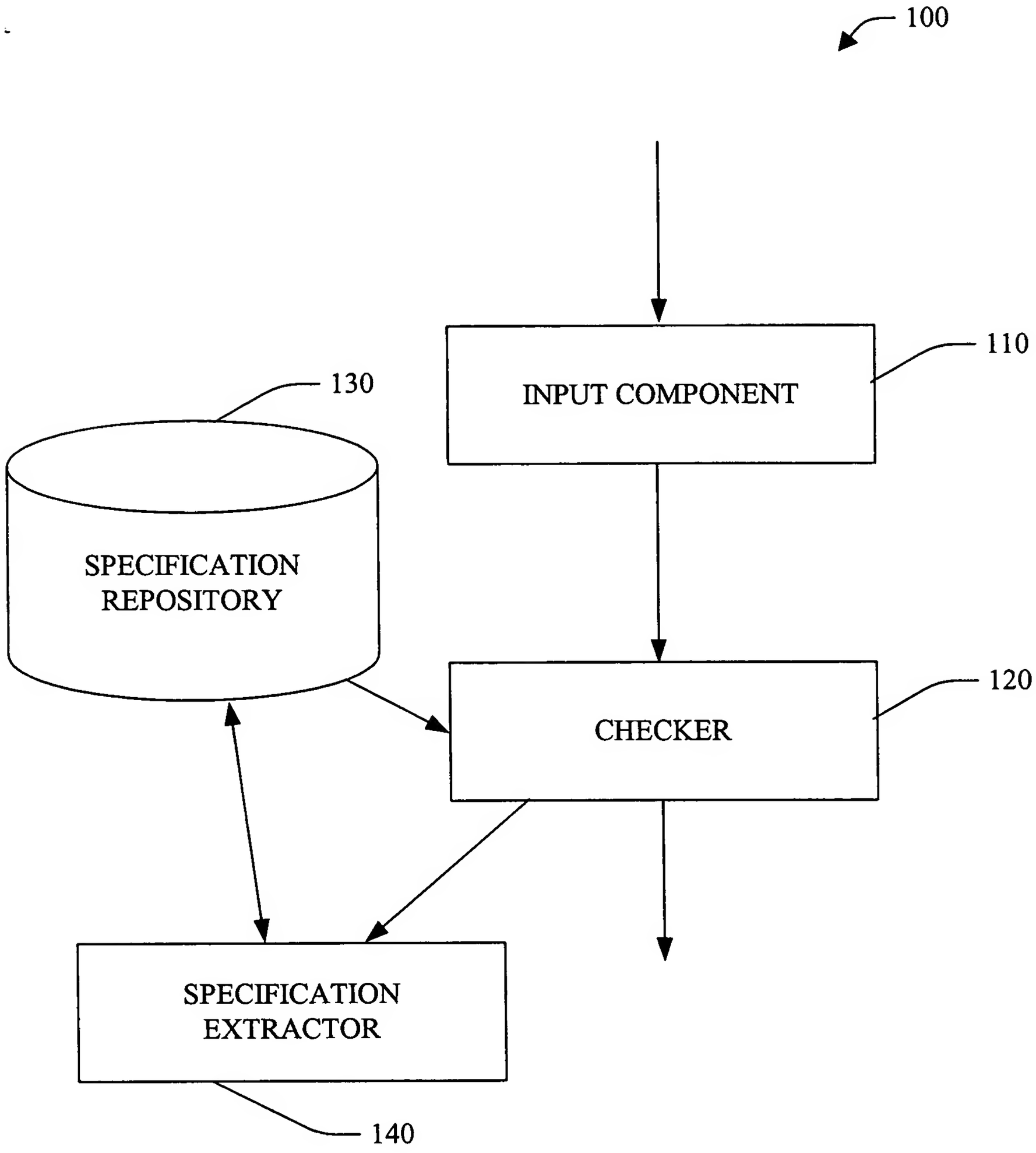
100
110
INPUT COMPONENT
130
SPECIFICATION
REPOSITORY
120
CHECKER
SPECIFICATION
EXTRACTOR
140

FIG. 13

1400

```
[WithProtocol( UnknownDB, KnownDB)]
class Publications : System.Web.UI.Page
{
  [InConnectionState(WhenEnclosingState=UnknownDB
    Status = ConnectionState.Closed,
    Host = AnyHost, Database = AnyDatabase)
   InConnectionState(WhenEnclosingState=KnownDB
    Status = ConnectionState.Closed,
    Host = XXX, Database = YYY )
  private SqlConnection m_sqlCn;

[ChangesState( UnknownDB , KnownDB )]
private void OnPageLoad (EventArgs e)
{
  m_sqlCn = new SqlConnection(...);
   //...
}

[InState( KnownDB )]
void WriteTRDetail ()
{
    m_sqlCn.Open();
   SqlCommand objCommand =
       new SqlCommand("EXEC ...", m_sqlCn);
   SqlDataReader objDataReader =
     objCommand.ExecuteReader();
   // ...
}
}
```

FIG. 14

1500

```
string GetPersonWebURL (
  [ InReaderState(
    ColumnNames = - "internalurl", "externalurl" ",
    ColumnTypes = - "nchar", "nchar" " ]
  SqlDataReader dr )
{
 if (dr["internalurl"] = = null)
    if (dr["externalurl"] = = null)
       return "";
    else
     // ...
}
```

FIG. 15

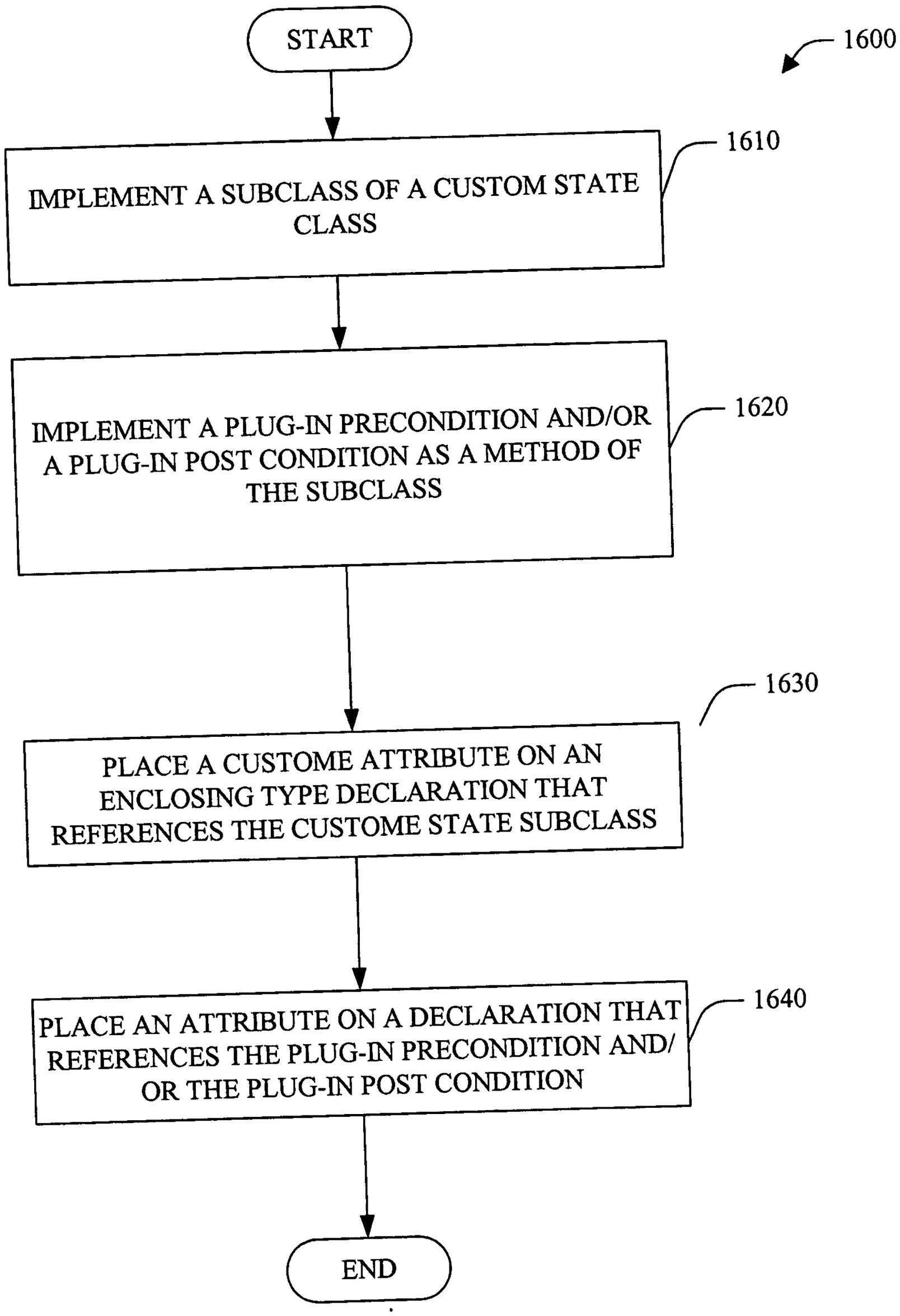
START
1600
IMPLEMENT A SUBCLASS OF A CUSTOM STATE CLASS
1610
IMPLEMENT A PLUG-IN PRECONDITION AND/OR A PLUG-IN POST CONDITION AS A METHOD OF THE SUBCLASS
1620
PLACE A CUSTOME ATTRIBUTE ON AN ENCLOSING TYPE DECLARATION THAT REFERENCES THE CUSTOME STATE SUBCLASS
1630
PLACE AN ATTRIBUTE ON A DECLARATION THAT REFERENCES THE PLUG-IN PRECONDITION AND/ OR THE PLUG-IN POST CONDITION
1640
END

FIG. 16

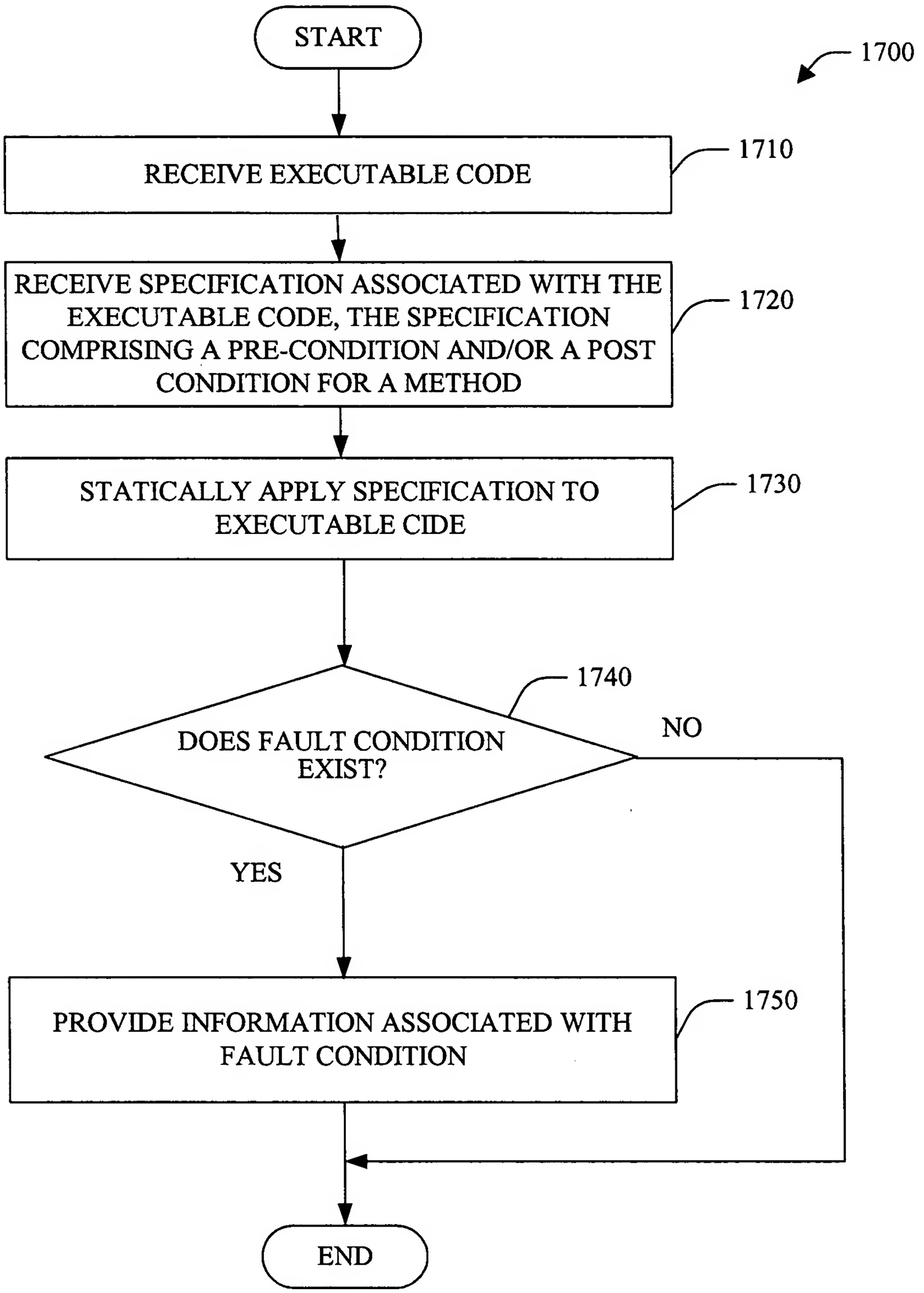
START
1700
RECEIVE EXECUTABLE CODE
1710
RECEIVE SPECIFICATION ASSOCIATED WITH THE EXECUTABLE CODE, THE SPECIFICATION COMPRISING A PRE-CONDITION AND/OR A POST CONDITION FOR A METHOD
1720
STATICALLY APPLY SPECIFICATION TO EXECUTABLE CIDE
1730
DOES FAULT CONDITION EXIST?
1740
NO
YES
PROVIDE INFORMATION ASSOCIATED WITH FAULT CONDITION
1750
END

FIG. 17

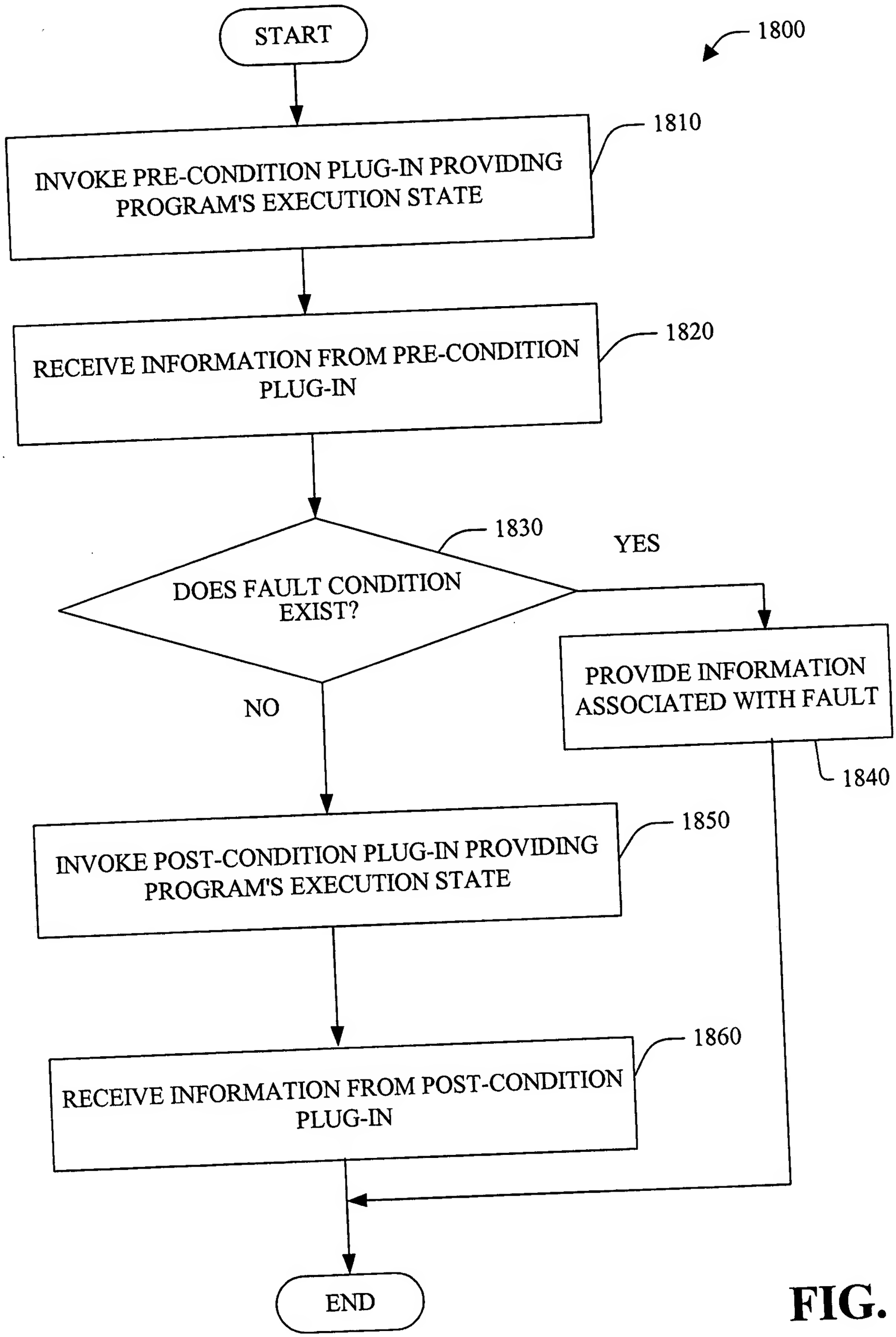
START
1800
INVOKE PRE-CONDITION PLUG-IN PROVIDING PROGRAM'S EXECUTION STATE
1810
RECEIVE INFORMATION FROM PRE-CONDITION PLUG-IN
1820
DOES FAULT CONDITION EXIST?
1830
YES
NO
PROVIDE INFORMATION ASSOCIATED WITH FAULT
1840
INVOKE POST-CONDITION PLUG-IN PROVIDING PROGRAM'S EXECUTION STATE
1850
RECEIVE INFORMATION FROM POST-CONDITION PLUG-IN
1860
END

FIG. 18

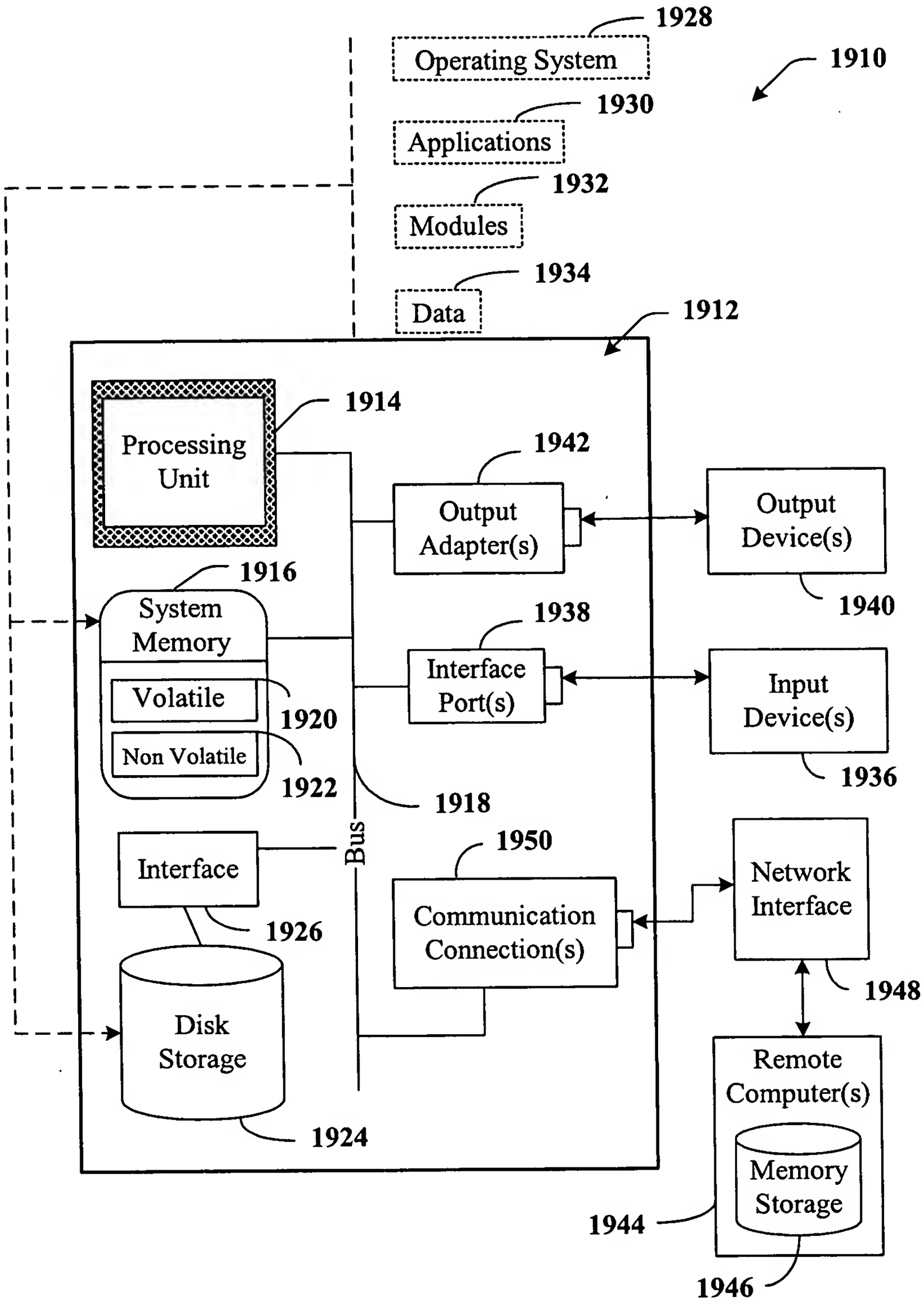
1928
Operating System
1910
1930
Applications
1932
Modules
1934
Data
1912
Processing Unit
1914
1942
Output Adapter(s)
Output Device(s)
1940
1916
System Memory
Volatile
1920
Non Volatile
1922
1938
Interface Port(s)
Input Device(s)
1936
1918
Bus
Interface
1926
1950
Communication Connection(s)
Network Interface
1948
Disk Storage
1924
Remote Computer(s)
Memory Storage
1944
1946

FIG. 19